Interface in Java

Prepared by
Rachit Joshi

Table of Contents

1. Introduction

2. What is an Interface?

3. Why Use Interfaces?

4. Features of Java Interfaces

5. Declaring an Interface

6. Implementing an Interface

7. Multiple Interface Inheritance

8. Default and Static Methods

9. Functional Interfaces

10. Advantages

11. Best Practices

12. Interview Questions

13. Conclusion

Introduction

Interfaces are one of the core concepts of object-oriented programming in Java. They define a contract that implementing classes must follow, enabling abstraction, loose coupling, and reusable code.

What is an Interface?

An interface is a reference **type in Java** that contains method declarations and constants. Classes implement interfaces to provide concrete behavior for the declared methods.

Why Use Interfaces?

- Achieve abstraction
- Support multiple inheritance of type
- Promote loose coupling
- Improve code maintainability
- Enable polymorphism

Features of Java Interfaces

Feature	Description
Abstract Methods	Methods implemented by classes
Default Methods	Provide implementation in interfaces
Static Methods	Called using interface name
Constants	Fields are public static final
Multiple Inheritance	A class can implement multiple interfaces

Declaring an Interface

```
interface Animal {
   void sound();
}
```

Implementing an Interface

```
class Dog implements Animal {
   public void sound() {
      System.out.println("Bark");
   }
}
```

Multiple Interface Inheritance

A Java class can implement more than one interface, allowing it to inherit behavior contracts from multiple sources.

Default and Static Methods

Since Java 8, interfaces can define default and static methods, enabling backward compatibility while adding new functionality.

Functional Interfaces

A functional interface contains exactly one abstract method and is commonly used with lambda expressions and the Stream API.

Advantages

- Encourages modular design
- Supports polymorphism
- Improves flexibility
- Enhances code reusability
- Simplifies testing

Best Practices

Keep interfaces focused on a single responsibility, use meaningful names, avoid unnecessary methods, and prefer composition where appropriate.

Interview Questions

- What is an interface in Java?
- Can an interface have constructors?
- What is the difference between an abstract class and an interface?
- What are default methods?
- Can a class implement multiple interfaces?

Conclusion

Interfaces are fundamental to Java application design. They define clear contracts between components, improve maintainability, and enable flexible, scalable object-oriented solutions.